REMARKS

In the Office Action, the Examiner rejected the claims 1-48 under 35 USC §102 and 35 USC §103. In response to the rejection under 35 USC §101, the claims have been amended to correct typographical errors and to further clarify the subject matter regarded as the invention. Applicant reserves the right to reintroduce claims of the same or similar scope in a subsequent continuation application. In response to the double patenting rejection, Applicant submits a terminal disclaimer.

Claims 35 and 38 have been cancelled. Claims 1-34, 36-37, and 39-48 remain pending. The claim rejections are fully traversed below.

Reconsideration of the application is respectfully requested based on the following remarks.

REJECTION OF CLAIMS UNDER 35 USC §102 AND 35 USC §103

In the Office Action, the Examiner rejected claims 1-30, 34-41, and 43-48 under 35 USC §102 as being anticipated by Judge et al, U.S. Patent No. 6,430,570, ('Judge' hereinafter). This rejection is fully traversed below.

It is important to note that many of the pending claims enable an application to have some input or control over its own execution according to an application lifecycle. For instance, the application itself may communicate with the application manager managing its execution. In this manner, the application may control (e.g., initiate or prevent) its own state change. Judge fails to disclose or suggest such a system.

Judge discloses a Java Application Manager for an Embedded Device. See Title. The application manager allows remote control of loading, starting, stopping, unloading, and application state querying of applications on an embedded device. See Abstract. In order to provide these capabilities over a network, the embedded device includes a network interface implementing a network protocol which allows clients to send requests to the Application

Manager of the embedded device. The Application Program Interface (API) provides remote devices with the ability to specify the desired management of the device. See col. 4, lines 24-37.

With respect to claims 1, the application itself can communicate a state change request to request a change in its own state. For instance, the application itself can communicate that it has decided to terminate and requests that the application enter a destroyed state (e.g., from a loaded state, a paused state, or an active state), as recited in claim 2. As another example, the application can communicate that the application has decided to pause its execution and requests that the application enter paused state (e.g., from the active state), as recited in claim 2.

Similarly, claims 3-4 enable an application to request that its own execution be resumed and its state be changed from paused to active. The application manager may then initiate execution of the application in response to the resume request. In this manner, the application can request that its own state be changed by the application manager.

In contrast, col. 9, lines 23-30 and col. 4, lines 24-37 of Judge indicate that such requests may be received from a client to change the state of another application. In fact, the Examiner refers to page 12, 4th paragraph, page 13, 3rd – 4th paragraphs, and more specifically to a "startAppl() request." As disclosed in this text and shown in the corresponding figures, the request is received from client 12, not the application 510. Thus, the state change of the application is managed by the application manager, with no input from the application itself. The input from the client does not enable the client to change its own state, nor does it enable the application to change its own state. In no manner does Judge disclose or suggest the ability of an application to communicate that it has decided to terminate, pause or resume its own execution. Accordingly, Applicant respectfully submits that claims 1-4 are patentable over the cited art.

Claim 5, as amended, enables the state of an application to be conditionally changed from a first state to a second state (e.g., terminated). In other words, the request is a conditional request that is conditional upon the application's decision to change from the first state to the second state. Specifically, claim 5, as amended, recites in part,

"requesting a first time that the application change its state from a first state to a second state by sending a request to the application, wherein the request is a conditional request that is conditional upon the application's decision to change from the first state to the second state, thereby enabling the application to allow or prevent its own state change from the first state to the second state in response to the conditional request;

determining whether the application has decided to allow its own state change from the first state to the second state in response to the conditional request by ascertaining whether the application has changed its state from the first state to the second state;"

If the application has not changed its state, the application manager may again request that it change its state from the first state to the second state (e.g., when a predetermined condition is satisfied).

In the Examiner's rejection of claim 5, the Examiner cites col. 13, lines 18-35, of Judge, which indicates that the Application Manager locates the correct application object, then calls the stopAppl() method on the Application. The Application object locates the ApplBase instance by the name passed into stopAppl() and calls its terminate() method. In other words, Judge merely discloses passing the name as a parameter. In other words, the stopAppl() method is called to terminate a particular application. However, the parameter does not indicate whether the termination is conditional or unconditional. Stated another way, the Application Manager of Judge does not enable the application to have any control over whether the application terminates in response to the execution of the stopAppl() method. Accordingly, Applicant respectfully asserts that claims 5-9 are allowable over the cited art.

Similarly, claim 10, as amended, enables the state of an application to be conditionally changed from a first state to a second state (e.g., terminated). If the application has not changed its state, the application is requested to change its state from the first state to another, third state. Judge fails to disclose or suggest such a process. Accordingly, Applicant respectfully submits that claims 10-14 are patentable over the Judge.

Claim 15 enables two subsequent state change requests to be submitted to an application when the first conditional state change request fails. For instance, with respect to claim 19, the application may reject the requested state change. Moreover, with respect to claim 20, the application may be unable to perform the requested state change. Judge fails to disclose or suggest such a process. Accordingly, Applicant respectfully submits that claims 15-20 are patentable over the cited reference.

Claim 21, as amended, recites a rule-based system for managing execution of an application according to an application lifecycle, which includes a mechanism for enabling the application to at least one of initiate and prevent its own state change from a first one of the plurality of states to a second one of the plurality of states. For instance, with respect to claim 27, an application environment object enables an application to communicate with the

application manager, thereby enabling the application to at least one of initiate its own state change, inform the application manager of the application-initiated state change, prevent its own state change, and inform the application manager that it is preventing its own state change that has been requested by the application manager. Claim 29 recites the application's ability to communicate its own state change to the application manager, wherein the state change has been initiated by the application.

Similarly, claim 30 recites the application's ability to request its own state change from the paused state to the active state. As set forth above, Judge fails to disclose or suggest a mechanism for enabling an application to communicate with an application manager, either in the form of informing the application manager of its state change or submitting a request for its state change. Claims 31-33 recite a display context, which is visible in a first state, and invisible in a second state (e.g., when in the active or paused state as recited in claim 32). As set forth above, Judge fails to disclose a display context as claimed. Accordingly, Applicant respectfully submits that claims 21-33 are patentable over Judge.

Claim 34 is directed to a digital television receiver, which loads an application when an application is present in data stream and executes the application according to an application lifecycle. In addition, claim 34, as amended, further recites a "means for enabling the application to prevent a change in state of the application requested by the application manager or to communicate to the application manager a state change of the application from one of a first set of the plurality of states to one of a second set of the plurality of states, wherein the state change of the application is initiated by the application."

Claims 36, 37 and 39 depend from claim 34. Claim 36, as amended, recites the ability of an application to pause or terminate its own execution. Similarly, claim 39, as amended, indicates that the application may initiate a change in its state from the active state to the paused state. In addition, claim 37 recites the ability of the application to initiate a change in its own state by requesting that the application manager change its state to a particular state, as claimed. Judge fails to disclose or suggest a mechanism for enabling an application to prevent or initiate its own state change. Accordingly, Applicant respectfully submits that claims 34-39 and 43 are patentable over Judge.

Claim 40 depends from claim 34, and recites the ability of an application to communicate to the application manager that it cannot change its state as the application manager has requested. Claim 41 depends from claim 40, and further recites enabling the application to raise

a state change exception indicating that the application cannot change its state as the application manager requested. Claim 42 recites enabling the application to raise a state change exception indicating that the application does not want to change its state as the application manager has requested, thereby enabling the application to prevent its own state change. Applicant respectfully asserts that Judge fails to disclose or suggest a mechanism for enabling an application to communicate with an application manager, as claimed, to enable the application to control its own state changes. Specifically, Applicant was unable to identify a state change exception indicating that the application does not want to change its state as the application manager has requested in col. 7, line 6 – col. 8, line 11, 25-36, as cited by the Examiner. Accordingly, Applicant respectfully submits that claims 40-42 are patentable over Judge.

Claims 44-46 each recite a class loader that is associated with a particular application. Claim 45 recites unloading classes, which may be performed by de-referencing the class loader as recited in claim 46.

Judge discloses a Java application manager for an embedded device. See Title. The application manager manages applications, where applications are cached in an application cache. See Abstract. The Application Manager unloads a class when requested using the unloadAppl() method or when the memory management handler selects the class to be unloaded as a result of a low- or no-memory condition. See col. 7, lines 27-36. By caching the class, the Application Manager maintains a reference to the class, thereby forcing the Java Virtual Machine not to garbage collect the class. See col. 7, lines 45-48.

Judge describes the Application Manager 24 as a "non-typical network based ClassLoader." See col. 3, lines 62-65. The Application Manager removes references to the objects that are chosen to be unloaded and causes the unreferenced objects to be removed. See col. 9, lines 41-51. In some JVM 22 implementations, the class's ClassLoader object may also have to be unreferenced for the class to be garbage collected. See col. 9, lines 41-51.

While reference is made in Judge to a ClassLoader object, Judge neither discloses nor suggests constructing (e.g., instantiating) a class loader for an application. Moreover, Judge neither discloses nor suggests constructing a class loader for each application to be executed. In other words, although a single class loader/Application Manager is disclosed, this class loader is not associated with an application to be executed. In fact, Judge implies that the Application Manager operates a single class loader for multiple applications. Thus, the Application Manager would not need to create one or multiple class loaders, as it operates independently to remove references to objects, as described above. In fact, FIG. 2 of Judge illustrates a single class loader

rather than a plurality of class loaders. Accordingly, Judge teaches away from an application manager that constructs a class loader for each application to be executed.

Moreover, Judge neither discloses nor suggests de-referencing the class loader (e.g., when execution of the application ends or when execution of the application is terminated), as recited in claim 46. Although Judge discloses dereferencing a ClassLoader object, Judge implies that the Class Loader object is associated with the Application Manager. In other words, Judge neither discloses nor suggests that the Application Manager de-references a class loader. In fact, Judge implies that the Application Manager/class loader is itself de-referenced. Thus, Judge neither discloses nor suggests an application manager dereferencing a class loader (e.g., when execution of the associated application ends or is terminated). Accordingly, Applicant respectfully submits that Judge fails to disclose an application specific class loader as claimed. Accordingly, Applicant respectfully submits that claims 44-46 are patentable over the cited reference.

Applicant believes that the independent claims and dependent claims are allowable for the reasons previously set forth. The dependent claims depend from one of the independent claims and are therefore patentable over the cited art for at least the same reasons. However, the dependent claims recite additional limitations that further distinguish them from the cited references. Hence, it is submitted that the dependent claims are patentable over the cited art. The additional limitations recited in the independent claims or the dependent claims are not further discussed as the above-discussed limitations are clearly sufficient to distinguish the claimed invention from the cited art. Thus, it is respectfully requested that the Examiner withdraw the rejection of the claims under 35 USC §102.

In the Office Action, the Examiner rejected claims 31-33 and 42 under 35 USC §103 as being unpatentable over Judge et al, U.S. Patent No. 6,430,570, ('Judge' hereinafter). This rejection is fully traversed below.

The Examiner admits that Judge does not disclose the display context being in a first state when the display context is visible and being in a second state when the display context is invisible. While the Examiner asserts that this would be obvious, Applicant respectfully asserts that this is impermissible hindsight. Applicant was unable to find a teaching or suggestion of this implementation in the cited reference.

With respect to claim 31, Judge fails to disclose or suggest a display context in a first state when the display context is visible and a second state when the display context is invisible. For instance, with respect to claim 32, which depends upon claim 31, Judge fails to disclose a

display context that is visible when the application is in an active state and invisible when the application is in a paused state. Moreover, with respect to claim 33, which depends from claim 31, Judge fails to disclose or suggest an application manager that determines the state of the display context according to one or more rules. Accordingly, Applicant respectfully submits that claims 31-33 are patentable over Judge.

Moreover, with respect to claim 42, the claimed invention enables the application to prevent its own state change that has been requested by the application manager. Judge fails to disclose or suggest such a system. Accordingly, Applicant respectfully submits that claim 42 is patentable over Judge.

If there are any issues remaining which the Examiner believes could be resolved through either a Supplemental Response or an Examiner's Amendment, the Examiner is respectfully requested to contact the undersigned attorney at the telephone number listed below.

Applicants hereby petition for an extension of time which may be required to maintain the pendency of this case, and any required fee for such extension or any further fee required in connection with the filing of this Amendment is to be charged to Deposit Account No. 50-0388 (Order No. SUN1P502).

Respectfully submitted,
BEYER, WEAVER & THOMAS, LLP

Elise R. Heilbrunn
Reg. No. 42,649

BEYER, WEAVER & THOMAS, LLP
P.O. Box 70250
Oakland, CA 94612-0250
(510) 663-1100